Universidad Autónoma de Madrid
Escuela Politécnica Superior

Proyecto fin de máster

UTILIZACIÓN DE TÉCNICAS DE CLUSTERING PARA MEJORAR LA DETECCIÓN DE META-TOPICS EN CONJUNTOS DE DATOS EXTRAÍDOS DE TWITTER

Máster en Ingeniería Informática

Carlos Delgado Calle
8 de septiembre de 2015

UTILIZACIÓN DE TÉCNICAS DE CLUSTERING PARA MEJORAR LA DETECCIÓN DE META-TOPICS EN CONJUNTOS DE DATOS EXTRAÍDOS DE TWITTER

AUTOR: Carlos Delgado Calle
TUTOR: Héctor Menéndez Benito
PONENTE: David Camacho Fernando

Applied Intelligence and Data Analysis
Dpto. de Ingeniera Informática
Escuela Politécnica Superior

Universidad Autónoma de Madrid
8 de septiembre de 2015

Resumen

Resumen

En los últimos años, las redes sociales no han dejado de cobrar importancia. En especial destaca Twitter, donde distintos usuarios comparten opiniones, noticias y todo tipo de contenido multimedia. El conjunto de datos generado en las redes sociales contiene una gran cantidad de información oculta a simple vista. Uno de los principales intereses de este trabajo es ordenar dicha información de cara a poder facilitársela a los usuarios. Para ello, este trabajo pretende evolucionar un modelo basado en meta-tópics, de tal forma que consiga una agrupación más precisa que la actual. El anterior modelo hacía uso del análisis semántico para detectar un único *meta-topic* a través de técnicas como LSA unidas a consultas semánticas a DBpedia, para obtener resultados con los que poder analizar la validez del modelo. En este trabajo se pretende mejorar dicho modelo dando un mayor peso a los valores obtenidos de DBpedia; dichos valores permiten saber, de forma certera, si un término se relaciona con el *meta-topic* deseado, pero en el diccionario no pueden encontrarse todos los términos; Por ello en este trabajo se presenta un modelo basado en el uso de la métrica NGD *(Normalized Google Distance)*, que calcula las distancias entre términos para posteriormente generar, utilizando técnicas de clustering, grupos de términos sobre los que poder propagar los valores de similitud obtenidos en DBpedia. Finalmente, se expondrán los resultados obtenidos con los dos modelos (el original y su evaluación) para comparar cual de los dos presenta mejores resultados.

Palabras Clave

Twitter, tweets, meta-topic, Topic Detection, DBpedia, LSA, análisis semántico, NGD, clustering

Abstract

Social Networks have become increasingly important over the last few years. One of the most relevant Social Network is Twitter. This Network allows users to share public comments and multimedia contents. Social networks contain large amounts of hidden information. This work aims to extract relevant data from Twitter and analyze and sort it in order to be able to provide relevant information to users. This projects wants to evolve a model based on meta-topics in order to improve the accuracy. The previous model used LSA techniques and DBpedia queries to obtain information who can be used to detect a *meta-topic*. Here, we improve the old model incrementing the weight of DBpedia values. Those values allow us to know whether a keyword belongs to a particular topic. We can not find all the keywords in the dictionaries. Hence, this work presents a new model where we calculate the distance between keywords via NGD *(Normalized Google Distance)* and, later, we apply clustering techniques to discriminate relevant groups. With this groups, we link the values via DBpedia with the rest of the keywords. Finally, we compare both, the results from the old and the new model, to show who presents the best results.

Key words

Twitter, tweets, meta-topic, Topic Detection, DBpedia, LSA, semantic analysis, NGD, clustering

Agradecimientos

Este trabajo está dedicado a mi familia, por estar siempre allí. A mi madre por apoyarme siempre, por muy oscuro que todo pareciese. A mis amigos por ayudarme a desconectar y recuperar fuerzas cuando la presión amenazaba con derrumbarme. A los compañeros del departamento, que me han acogido como si fuese uno más y en especial a Héctor por guiarme como nadie más habría sido capaz. Y por último agradecer a todo aquel que se digne a leer este proyecto, porque solo por vosotros ha merecido la pena el esfuerzo.

Índice general

Índice de figuras

Índice de tablas

1

Introducción

Twitter es un servicio de microblogging, o lo que es lo mismo, un servicio que permite a sus usuarios enviar y publicar mensajes breves (de hasta un máximo de 140 caracteres), compartiendo información de forma libre. Este tipo de servicios permite expresar opiniones breves, comentar otras o compartir contenido multimedia de diversa índole; dado que es utilizado por un amplio y diverso conjunto de usuarios de distinta edad, género y condición social, esto lo convierte en un gran nicho de información relevante sobre una gran cantidad de temas. En la actualidad, existen varios de estos servicios (muchos aparecen y desaparecen a diario), pero Twitter se mantiene desde su creación y lanzamiento en el año 2006 (por Jack Dorsey, Evan Williams, Biz Stone y Noah Glass) como uno de los más utilizados por los usuarios. Esta razón y su *API* pública han determinado que finalmente se haya adoptado como foco de información para el desarrollo del trabajo realizado.

Dado que cada usuario suele tener gustos variados (como deportes, videojuegos, política, cine y música, entre otros) y suele seguir a usuarios de distinta condición social (compañeros de clase/trabajo, atletas profesionales, amigos, empresas y compañías discográficas, entre otros), el proceso de extracción de información específica útil sobre los gustos del usuario se vuelve muy complejo.

En el modelo presentado en el trabajo **Modelo de Indetificación de Meta-Topics a Través de Análisis Semántico de Conjuntos de Datos Extraídos de Twitter [45]** se pretendía obtener las publicaciones de los gustos de los usuarios a los que sigue el sujeto y organizarlas de acuerdo a un tema general que comprendiera varios temas (o *meta-topic*) como podría ser deportes, videojuegos, cine, etc. dependiendo del usuario. Para ello, se llevaba a cabo un análisis semántico de las publicaciones (o tweets) extraídas de dichos usuarios, para su posterior análisis, de modo que se pudiera encontrar una relación entre los tweets y el meta-topic seleccionado (preseleccionado para este modelo). De este modo se podía generar un ranking de los tweets por cercanía al meta-topic.

Partiendo de los resultados obtenidos en dicho modelo, en este trabajo se presenta un nuevo modelo en el que se utilizan técnicas más modernas de análisis semántico (como la métrica de similitud *NGD* o técnicas sencillas de clustering de información) en un intento de mejorar la eficacia del anterior modelo a la hora de clasificar los *tweets* de los usuarios.

Una vez obtenida esta información, no sólo podría ser utilizada para facilitar la búsqueda por *meta-topics* al usuario, si no que permitiría, por ejemplo, generar grupos de usuarios que

compartiesen intereses comunes (recomendación en la web), así como analizar e investigar las tendencias de la Red Social en cada momento, las comunidades que se generan dentro de esta, etc. Estas investigaciones posteriormente pueden aplicarse a la hora de realizar campañas de marketing más efectivas, a mejorar la organización a nivel empresarial, a situar la presencia de las distintas marcas en el mercado, etc.

Para tratar estos temas se suelen usar técnicas de Text Mining (o Data Mining) [6] y de TDT (Topic Detection and Tracking) [6] que permiten obtener información a partir de los textos, como es el caso de **Relfinder** [39] y **Wikipedia Miner** [47]. Relfinder presenta de manera visual (mediante grafos) las relaciones obtenidas a través de un dataset (a menudo obtenido de Dbpedia) entre diversos términos; mientras que Wikipedia Miner está formado por un conjunto de herramientas creadas alrededor de Wikipedia, destinadas a extraer información relevante de ésta (búsqueda de términos, comparación semántica entre estos, desambiguación de temas, etc).

En este trabajo se aplican técnicas que buscan relaciones semánticas mediante el uso de diccionarios como Wordnet [46] o DBpedia [36] combinado con algoritmos capaces de relacionar términos entre sí, como LSA o métricas de similitud (como *NGD*). Tanto el modelo previo como el expuesto en estre trabajo poseen una arquitectura capaz de extraer los tweets deseados de Twitter y analizarlos semánticamete para, finalmente, devolver un ranking con respecto al *meta-topic* preseleccionado mediante este tipo de técnicas.

Al igual que en el desarrollo del anterior modelo, los problemas que se presentan son de diversa índole y se deben tener en cuenta las limitaciones le proyecto. Primero, se trata de un proyecto muy amplio y ambicioso y por ello es necesario marcar objetivos y acotarlo en distintos niveles para poder abarcarlo. Por otra parte, el análisis semántico nunca ha sido sencillo y exacto y por tanto los resultados distan mucho de ser completamente acertados (como puede ocurrir en otras ramas de la Inteligencia Artificial). El trato con información semi-estructurada impone un análisis preliminar de cada meta-topic que se quiere tratar, lo cual, sumado a lo dicho anteriormente ha desembocado en la selección de un único meta-topic a la hora de desarrollar estas primeras versiones del proyecto. Todo lo anteriormente comentado hace imposible crear una versión completa del proyecto, por lo que se ha optado por realizar modelos en los que se trata un único meta-topic, de modo que se obtengan unos datos concretos que se puedan analizar para, más adelante, poder desarrollar modelos más completos. Por último, cabe destacar que la arquitectura del proyecto ha sido diseñada de tal forma que sólo es necesario modificar el módulo de consultas a diccionarios (en el caso actual *DBpedia*) para poder adaptarlo a nuevos *meta-topics* o nuevas técnicas.

El resto del trabajo se encuentra estructurado de la siguiente forma: En la Sección 2 se introduce el estado del arte que rodea al proyecto. La Sección 3 describe la arquitectura y el funcionamiento de los modelos a comparar. En la Sección 4 se muestra la comparativa realizada entre ambos modelos y los resultados obtenidos. Finalmente, la Sección 5 incluye el análisis de los resultados en forma de conclusiones, así como un apartado dedicado al trabajo futuro.

1.1. Motivación del proyecto

Las redes sociales en formato digital se han extendido rápidamente, obteniendo mucha importancia a nivel social. La cantidad de información que contienen es inmensa y a día de hoy hay una carrera por ver quién consigue darle un mejor uso a dicha información. Ciertas comunidades incluso han comenzado a llamar al *Big Data* el *petróleo del siglo XXI*. La posibilidad de formar parte de dicha carrera es, sin lugar a dudas, la mayor motivación para seleccionar este trabajo a la hora de trabajar.

Por otro lado se ha optado por el análisis semántico y la selección de un *meta-topic*, porque se busca organizar la información de una forma cercana para el usuario, analizándola por su contenido y no tanto por su forma, por lo que el análisis semántico se vuelve indispensable.

1.2. Objetivos del proyecto

Los objetivos de este proyecto son los siguientes:

- **Extracción de tweets:** Extraer los tweets de Twitter mediante el uso de la API que este provee.
- **Extracción de keywords:** Analizar los tweets obtenidos para obtener las keywords que los forman.
- **Aplicación de métricas de análisis semántico a las keywords:** Se busca obtener relaciones semánticas de las keywords entre sí.
- **Análisis mediante consultas a DBbpedia de las keywords:** Aplicar consultas a DBpedia para obtener relaciones semánticas entre las keywords y el *meta-topic*.
- **Valoración final de las keywords:** Valoración basada en los datos obtenidos para las keywords en los objetivos anteriores.
- **Creación de un ranking de tweets:** Creación de un ranking con respecto al *meta-topic* seleccionado, tras aplicar una valoración a los tweets basada en las keywords que los conforman.
- **Comparativa del ranking:** Comparativa entre el antiguo modelo y el nuevo modelo presentado en este documento en pos de comprobar cual obtiene unos mejores resultados.

2

Estado del Arte

En este apartado se trata de forma general sobre qué es *Data Mining*, así como las secciones en las que tiende a dividirse y una explicación más detallada sobre algunas de estas secciones. También se tratarán los modelos de análisis más comunes y se explicarán diversas técnicas de validación y métodos de evaluación con respecto al análisis de redes sociales. Finalmente se expone una serie de herramientas útiles a la hora de llevar a cabo este tipo de análisis.

2.1. Data Mining

Según *Daniel T. Larose* Data Mining es "el proceso de descubrir nuevas y significativas correlaciones, patrones y tendencias, mediante la criba de grandes cantidades de datos almacenadas en repositorios, utilizando tecnologías de reconocimiento de patrones, así como técnicas matemáticas y estadística" [35]. Las técnicas aplicadas en Data Mining suelen dividirse en 5 fases:

1. **Extracción de datos:** Obtención del conjunto de datos que posteriormente será analizado.
2. **Preprocesamiento de datos:** Una vez extraídos, los datos son preparados para su posterior análisis. Este paso se divide en tres fases [35]: evitar la clasificación errónea, reducción de la dimensión (mediante proyecciones o técnicas de selección de características) y normalización del rango.
3. **Generación del modelo:** Conforma la parte más importante del análisis de datos. El objetivo del modelo es la búsqueda de patrones en los datos. Habitualmente se utilizan técnicas estadísticas o de *Machine Learning* para generar el modelo [35].
4. **Validación del modelo:** El proceso de validación depende en gran medida del tipo de modelo generado. Lo más común es utilizar la validación para clasificadores [35].
5. **Aplicación del modelo:** Una vez generado y validado el modelo su objetivo es ser aplicado con algún objetivo, por ejemplo, predecir el comportamiento de nuevas entradas de datos.

La aplicación del *Data Mining* difiere según los campos donde se aplique y dependiendo de la representación de los datos. La aplicación más habitual es el análisis directo de los datos, comúnmente aplicada a datos numéricos (se pueden encontrar algunos datasets en el UCI Machine Learning Repository [2]). Otro campo de aplicación está más centrado en el análisis de imágenes, donde se utilizan algoritmos como la segmentación de imagen [22], detección de objetos [40], el reconocimiento facial [59] o la reconstrucción 3D [56] entre otros. Con el auge de las redes sociales, en la actualizad existen diversos métodos de *Data Mining* orientados al análisis de las mismas [1]. Algunas de estas técnicas se basan en las redes que generan los propios usuarios, mientras que otras analizan la información intercambiada dentro de las mismas. Muchos de estos modelos son extraídos del análisis de redes complejas, como se verá más adelante.

2.2. Extracción de datos

A la hora de extraer los datos a analizar existen varias alternativas. Existen repositorios o bases de datos públicos de donde obtener datasets con los que probar los modelos. Uno de los más usado en los trabajos de *Data Mining* es el UCI Machine Learning Repository [2]. Pero dependiendo de la aplicación que se quiera dar al modelo, también dependerá el origen de los datos.

Los datos que se van a analizar en este trabajo provienen de la red social Twitter, donde la gente normalmente comparte públicamente información sobre sus opiniones personales. Está dividida en dos tipos de comportamiento del usuario: follower y following. Como follower, el usuario recibe información de los usuarios a los que el sigue (following). Como following, el usuario envía información a sus seguidores (o followers). La información que el usuario comparte se denomina *Tweet* y está limitada a 140 caracteres, que pueden contener información sobre opiniones personales, fotos, links, etc. Un usuario puede también hacer *re-tweet* de la información de otro usuario y así compartirla. Cabe destacar que existen aplicaciones para el análisis de las Redes Sociales, que permite a los investigadores extraer la información de Twitter (como Twitter API [42]) o Facebook (Facebook API [26]), entre otros.

2.3. Preprocesamiento de datos

El proceso de preprocesado de datos consiste en una serie de pasos orientados a simplificar la información contenida en los datos, con el objetivo de homogeneizar y facilitar su posterior análisis. Cuando se trabaja con datos estructurados como documentos, como es el caso de este trabajo, el preprocesado se divide en tres pasos:

1. Primero se eliminan las Stop-Words (artículos, pronombres, preposiciones, etc) y los caracteres especiales de los textos de los documentos.
2. A continuación se extraen las *keywords* de cada documento, generando una matriz término-documento.
3. Por último se aplican técnicas de selección de características para elegir las *keywords* más relevantes para el análisis, reduciendo así el espacio de búsqueda sobre el que se trabaja.

La información extraída de los datos debe ser inicialmente analizada y almacenada en algún tipo de sistema de almacenaje o base de datos. Este proceso también es utilizado para evitar outliers (valores atípicos), errores en la clasificación y pérdida de información. Métodos como la correlación estadística o los histogramas se utilizan para limpiar el dataset, reduciendo así el

número de variables [35]. En la reducción de la dimensión de un dataset son frecuentes las proyecciones, sin embarco, métodos como PCA (Principal Component Analysis) o LDA (Lineal Discriminant Analysis) no terminan de ofrecer una perspectiva completa del problema [16]. Estos métodos generan nuevas variables estimadas mediante componentes principales o proyecciones lineales, tratando de separar los datos y reducir su dimensión. Estas técnicas tienden a deshacerse de la información original en el proceso de la proyección, lo cual aumenta la dificultad de la interpretación humana a la hora de analizar los resultados. Es por ello que es preferible evitarlas.

Existen varias técnicas que reducen las dimensiones evitando el uso de proyecciones. Aplican una búsqueda guiada que busca las variables más útiles para el análisis entre todos los atributos presentes. Estos métodos se conocen como "métodos de selección de características" [33].

Las aplicaciones de estos métodos se realizan desde perspectivas muy diversas, por ejemplo, Curiel et al. [14] aplica algoritmos genéticos para simplificar el pronóstico de la endocarditis utilizando una codificación donde cada individuo de la población se basa en un conjunto de características. Blum and Langley [7] muestra algunos ejemplos de selecciones de características relevantes en diferentes datasets, aplicadas a diferentes técnicas de aprendizaje automático. Los autores definen distintos grados de características relevantes, como fuertes o débiles. También se estudian metodologías como la búsqueda heurística, aproximaciones de *filters and wrapper* que son métodos de selección de características automáticas generalmente validados mediante técnicas de clasificación. Algunas de estas técnicas introducen *over-fitting* al modelo, lo que reduce su fiabilidad. En Roth and Lange [55] se aplican estas técnicas para los problemas de clustering.

El último paso del preprocesado de datos es la normalización de los mismos. Esto permite comparar entre sí características con distintos rangos de valores. Los métodos de normalización Z-Score [9] y Min-Max [28] son los más comunes. Ambos buscan y llevan los atributos a un rango concreto. Min-Max tiene un rango fijo, [0,1] (sensible a outliers), mientras que Z-Score depende de la media y la desviación estándar (aproxima la distribución a una distribución normal, se utiliza por lo general para evitar outliers). Estos algoritmos obtienen sus valores normalizados de aplicar las siguientes ecuaciones:

- **Min-max:** Calcula los valores máximos y mínimos de los atributos aplicando:

$$x' = \frac{x - min(X)}{max(X) - min(X)}$$

- **Z-Score:** Calcula la desviación media y estándar de los valores aplicando:

$$x' = \frac{x - mean(X)}{SD(X)}$$

Una vez que los datos están listos para el análisis, la generación del modelo comienza. Este trabajo se basa en técnicas de aprendizaje no supervisado.

2.4. Modelos de Análisis

En esta sección se describen algunos de los modelos más clásicos de *Data Mining*, así como una breve introducción a las Redes Complejas. Todo ello desde una perspectiva sobre el análisis de Redes Sociales.

2.4.1. Modelos Clasicos de Data Mining

Las técnicas de *Machine Learning* que se utilizan en *Data Mining* son principalmente técnicas de Clasificación y Clustering [35]. Las técnicas de clasificación buscan patrones dentro del conjunto de datos de forma supervisada, es decir, utilizan datos ya etiquetados para generar los modelos [35]. Las técnicas de clustering buscan los patrones de forma ciega, sin un etiquetado previo, y generan los modelos a partir de métodos estadísticos [35].

Dentro de las técnicas de clasificación clásicas destacan:

- **Árboles C4.5** [52]: Es la técnica más clásica en clasificación. Divide los datos de forma lineal utilizando límites en los atributos, generando, así, un árbol de decisión. La división se elige utilizando una métrica como la entropía de datos.
- **Naive Bayes (NB)** [18]: El clasificador considera cada característica como indepediente del resto. Cada una de ellas contribuye a la información del modelo. Se basa en la Ley de Probabilidad de Bayes
- **Algoritmo K-Nearest Neighbour (KNN)** [13]: Clasifica los elementos de acuerdo con sus vecinos. Dependiendo del valor de K, considera los K-vecinos más cercano para estimar el valor de una nueva instancia no clasificada.
- **Support Vector Machines (SVM)** [12]: Cambia la dimensión del espacio de búsqueda a través de diferentes funciones de kernel (o núcleo), que tratan de mejorar la clasificación. Estas funciones llevan los datos a un espacio ampliado donde se utiliza un hiperplano para separar los datos de forma lineal.

Dentro de las técnicas de clustering destacan:

- **K-means** [41]: Dado un número fijo de clusters, K-means trata de encontrar una división del conjunto de datos basado en un conjunto de características comunes dadas por las distancias o las métricas que se utilizan para determinar cómo debe definirse el cluster. En el caso de K-means cada cluster está representado por un centroide al que los datos más cercanos se asocian.
- **Expectation-Maximitation (EM)** [48]: Es un método de optimización iterativo que calcula algunos parámetros desconocidos calculando las probabilidades de pertenecer al cluster utilizando una o varias distribuciones de probabilidad. Su objetivo es maximizar la probabilidad global de que los datos estén en los clusters finales. Para ello va modificando los parámetros de las distribuciones hasta que el modelo se adapta a los datos. Esta adaptación se mide con la verosimilitud.
- **Clustering por threshold**: Método basado en distancias. Preestableciendo un threshold este método es capaz de agrupar en clusters los elementos. Compara el valor del elemento con el del threshold, si este supera al threshold se considera que los elementos son similares, agrupándolos en un mismo cluster. Una vez terminado el proceso se subsumen aquellos clusters que están contenidos en otros de mayor tamaño.

Dentro del conjunto de aplicación de estos modelos, este trabajo se centra especialmente en los modelos de *Text Mining* [6]. Estos modelos utilizan documentos para aplicarles técnicas de clustering y clasificación [6], entre otras, de cara a agrupar documentos y clasificarlos por similitud, identificar términos (TDT) y buscar tendencias en los textos, como se hace en las redes sociales.

2.4.2. Redes Complejas

El análisis de las redes complejas se ha convertido en un campo muy importante, especialmente en física. Una de sus principales aplicaciones dentro del análisis de datos es el análisis de Redes Sociales, que son representadas normalmente por Redes Complejas. Existen cuatro tipos básicos de Redes: **Random Network** [21], **Regular Network** [60], **Scale-Free Network** [4] y **Small World Network** [58].

El análisis de una Red Social Compleja puede llevarse a cabo mediante algoritmos como PageRank [8] y HITS [32]. Ambos toman información sobre los nodos más representativos de la red y cómo esto afecta a la red en general. PageRank es un algoritmo que analiza los enlaces ente nodos, utilizado inicialmente por el motor de búsqueda web de Google. Hyperlink-Induced Topic Search (HITS) también conocido como "hubs y autoridades", es un algoritmo de análisis de enlaces que clasifica las páginas Web, desarrollado por Jon Kleinberg. Fue un precursor de PageRank.

Por otro lado, también es destacable la búsqueda de comunidades en las redes sociales. Una comunidad puede ser considerada como un subconjunto de individuos con conexiones relativamente fuertes, intensivas y directas entre ellas[24]. Algunos algoritmos que se centran en abordar este problema mediante un proceso determinista son *Edge Betweenness Centrality (EBC)* [25] y *Clique Percolation Method (CPM)* [17].

Otra aproximación relacionada con la búsqueda de comunidades puede encontrarse en [54], donde se utilizan diferentes mecanismos estadísticos para la detección de la comunidad. Los algoritmos genéticos también se aplican a la búsqueda de comunidades o clusters. Entre ellos se encuentran algoritmos genéticos de aglomeración [38] y algoritmos evolutivos multi-objetivo [31].

2.5. Validación del Modelo

La validación de los modelos es en general muy variada. En este trabajo nos hemos enfocado en una evaluación basada en las *curvas ROC (Receiver Operating Characteristic)*. Para poder definirlas, es necesario definir los siguientes conceptos en relación a cómo una instancia ha sido correcta o incorrectamente clasificada:

- True-Positive (TP): La instancia ha sido clasificada correctamente como parte de la clase.
- False-Positive (FP): La instancia ha sido incorrectamente clasificada como parte de la clase.
- True-Negative (TN): La instancia ha sido clasificada correctamente como externa a la clase.
- False-Negative (FN): La instancia ha sido incorrectamente clasificada como externa a la clase.

A continuación se definen los conceptos con los que las *curvas ROC* trabajan:

Sensibilidad: indica la capacidad de nuestro estimador para dar como casos positivos aquellos casos que realmente lo son.

$$Sensibilidad = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{2.1}$$

Especificidad: indica la capacidad de nuestro estimador para dar como casos negativos aquellos casos que realmente son negativos.

$$Especificidad = \frac{TN}{N} = \frac{TN}{FP + TN} = 1 - \frac{FP}{N} \tag{2.2}$$

Una *curva ROC* se define como una representación gráfica de la *Sensibilidad* frente a (1 - *Especificidad*) para un sistema de clasificación binario, según se varía el umbral de discriminación. El análisis de la *curva ROC* proporciona herramientas para seleccionar aquellos modelos que son posiblemente más óptimos y descartar aquellos modelos que sean subóptimos.

Un espacio ROC se define por (1 - Especificidad) y Sensibilidad como ejes x e y respectivamente, y representa los intercambios entre *True Positives* y *False Positives*. Cada resultado de predicción representa un punto en el espacio ROC.

El mejor método posible de predicción se sitúa en la esquina superior izquierda del espacio ROC, representando un 100 % de Sensibilidad y un 100 % de Especificidad. Por el contrario, una clasificación completamente aleatoria daría un punto a lo largo de la línea diagonal o *línea de no-discriminación*, desde el extremo inferior izquierdo hasta la esquina superior derecha. Un ejemplo de método aleatorio sería la predicción de cara o cruz de lanzar una moneda. Según aumentase la muestra el clasificador aleatorio se desplazaría a la posición (0.5, 0.5) del espacio ROC.

En la *Figura 2.1* podemos ver el espacio ROC representado de manera visual. Un clasificador aleatorio se encontrará situado en la línea diagonal, mientras que si clasifica de forma positiva se encontrará por encima de dicha línea y si clasifica de forma erronea aparecerá representado por debajo de dicha diagonal.

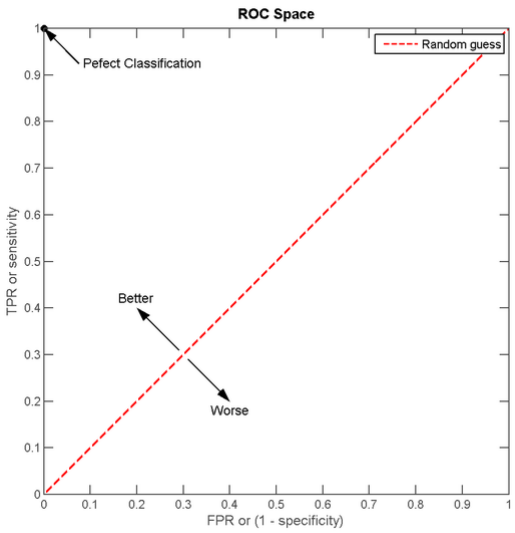

Figura 2.1: Espacio ROC

2.6. Aplicaciones en el Análisis de Redes Sociales

Desde la aparición de la Web 2.0 en 1999 el *Análisis Social* es uno de los campos donde más se trabaja. En este tipo de sitios web se genera una gran cantidad de información entre usuarios (originalmente usuarios de foros y chats). Fue en 2004 cuando Mark Zuckerberg fundó Facebook, una de las Redes Sociales más relevantes hasta la fecha, donde se permite compartir comentarios y opiniones entre usuarios. Dos años más tarde, en 2006, Jack Dorsey creó Twitter. Esta Red Social, junto a los servicios de Migroblogging, es una de las más famosas y más utilizada plataforma para el análisis de datos sociales o *Social Networks Analysis*. En esta sección se describen algunas de las aplicaciones de *Data Mining* en algunas de las redes sociales más populares.

2.6.1. Facebook

Facebook es una de las redes sociales más importantes en la actualidad. Originalmente diseñada para intercambiar fotos entre usuarios que eran amigos entre sí dentro de la red, hoy en día se utiliza para compartir todo tipo de contenido multimedia. Las características más relevantes de Facebook son:

- Estructura de *Amistad*: los Usuarios pertenecen a una comunidad de amigos formada por gente de su entorno social.
- Opción de *Me gusta* (o *Like*): expresa el interés de distintos usuarios por contenido multimedia publicado por otros usuarios o por ellos mismos.
- Opción de *Compartir*: permite a un usuario publicitar una publicación en su perfil, al contrario que la opción de *Like*, esta opción puede tener connotación negativa, aunque normalmente es una forma de difundir un contenido que se considera de interés.
- La opción de *Comentar*: permite a los usuarios comentar cualquier cosa (incluso otros comentarios), generando así interacciones entre ellos.

Usando la estructura de *Amistad* como punto de partida es posible analizar la red generada. El análisis de la red puede estar enfocada desde distintos puntos de vista, por ejemplo, en [23] se discuten las características mesoscópicas de la estructura de la comunidad de esta red, después de que se diera a conocer que las comunidades representan las unidades de agregación entre los que los usuarios se reúnen e interactúan.

En [10] centran el trabajo en participantes de las Redes Sociales online. La información se organiza como un grafo no dirigido y es anónima. Desarrollaron un set de herramientas para analizar propiedades específicas, como el grado de la distribución, medidas de centralidad, leyes de escalada y la distribución de la amistad entre usuarios.

En [3] se enfrentan a un problema de predicción de links. Dada una instantánea de una red, infieren qué interacciones entre miembros existentes son propensas a ocurrir en el futuro cercano o qué interacciones existentes estamos perdiendo.

Finalmente, en [37] se introduce un nuevo dataset público basado en manipulaciones de Facebook. En la segunda mitad de este *paper* se utiliza un algoritmo de búsqueda en la comunidad para encontrar los subgrupos definidos por género, raza / origen étnico y datos socioeconómico.

2.6.2. Twitter

Twitter es una Red Social conocida por compartir contenido multimedia y crear conversaciones con mensajes de menos de 140 carateres, creando un gran nicho de información. Dicha información realizada mediante el intercambio de opiniones, puede ser analizada.

En [57] se presenta un trabajo donde se examina la influencia de la distancia geográfica, las fronteras, el lenguaje y la frecuencia de vuelo entre distintas zonas. En los resultados pueden observarse los lazos sociales de la gente que reside en las mismas zonas metropolitanas, y que entre las agrupaciones regionales, la distancia, las fronteras nacionales y las diferencias idiomáticas se pueden predecir los lazos sociales presentes en Twitter.

En [30] se analiza el uso de Twitter. Para ello, presentan una taxonomía sobre la caracterización de las intenciones de los usuarios a la hora de realizar publicaciones. Sumando las intenciones de los usuarios en datos extraídos de distintas comunidades se muestra qué usuarios con intenciones similares tienden a conectar entre ellos.

En [44] se propone un método para analizar la Red Social de Twitter, donde se toma una instantánea de los extremos de la red y los tiempos de creación de las cuentas de usuario para inferir con precisión cuándo se formaron estos extremos. Este método puede ser exacto en la teoría y ellos demostraron empíricamente que para un gran subconjunto de relaciones en Twitter tenía una precisión de unas pocas horas en la práctica.

Finalmente, en [34] se estudia la topología de las características de Twitter y su capacidad como un nuevo medio de compartir información. Los resultados presentan una distribución *non-power-law* para los *followers*, un diámetro efectivo relativamente corto y una reciprocidad baja. De cara a identificar usuarios influyentes en Twitter se presenta un ranking basado en el número de *followers* y otro basado en *PageRank*, encontrando que ambos rankings son bastante similares.

2.6.3. Perspectiva Semántica

Uno de los enfoques más actuales del análisis de datos está aplicado a la comprensión semántica. En este campo destaca sobre todo la web semántica [29]; Un enfoque reciente utilizado para proporcionar nuevas formas de uso de la web. Este enfoque está basado en las relaciones semánticas existentes entre los diferentes conceptos que se pueden encontrar en Internet. Con esta información, se trata de mejorar la experiencia del usuario a través de la mejora de los sistemas de acceso a la información.

Dbpedia [36] destaca dentro de las bases de datos semánticas. Es una base de conocimiento representada mediante relaciones semánticas. Utiliza varios motores. La combinación de los conocimientos extraídos de dichos motores proporciona una herramienta como base de conocimientos que ayuda a nuestra arquitectura a trabajar. Dbpedia representa cada concepto como un documento o página, y vincula cada página con las demás de acuerdo a ciertas relaciones, por ejemplo, Oscar Wilde *nació en* Dublín, por lo tanto la página de Oscar Wilde estará vinculada con la página de Dublín. DBpedia también proporciona categorías para agrupar los diferentes conceptos. Cada concepto pertenece a una o varias categorías. Desde el punto de vista de la Web Semántica, también proporciona ontologías sobre diferentes temáticas (la ontología es una rama de la ciencia que estudia 'lo que hay').

Por último cabe mencionar una de las mejores herramientas para la web semántica es SPARQL [51]. SPARQL es un lenguaje de consultas estructurado de manera similar a SQL, que puede ser utilizado con sistemas tales como DBpedia. El módulo de consultas a DBpedia presentado en este trabajo se ha diseñado utilizando una estructura en forma de árbol con

consultas SPARQL a distinto nivel para interactuar con DBpedia y determinar si un *keyword* pertenece o no a un determinado *meta-topic*.

2.7. Herramientas Software

Existe una gran variedad de herramientas utilizadas en procesos de *Data Mining* y el análisis de Redes Sociales o *Social Data Mining*. A continuación se exponen algunas de las herramientas más relevantes:

- **Gephi** [1]: Gephi [5] es un software de visualización y análisis de grafos orientado a todo tipo de redes y sistemas complejos y grafos dinámicos y jerárquicos. Dispone de varios algoritmos implementados para el análisis de Redes Sociales, como *PageRank*, *HITS*, etc. También presenta algoritmos para mejorar la visualización del grafo, utilizando diferentes *layouts* y es capaz de calcular diferentes métricas del grafo como su grado (*power-law*), *betweenness*, *closeness*, densidad, longitud de la trayectoria, diámetro, modularidad o coeficiente de clustering.
- **Graphviz** [2]: Graphviz [20] (Graph Visualization Software) es un software *open source* de visualización de grafos. Puede representar diagramas, grafos y redes. Se aplica para llevar a cabo análisis de redes, bioinformátia, ingeniería del software, diseño de webs, *machine learning* y puede llevar a cabo visualizaciones para otros dominios técnicos.
- **JUNG** [3]: JUNG [49] (the Java Universal Network/Graph Framework) es una librería Java que provee de algunas herramientas para el modelado de grafos y redes, así como su análisis y visualización. Ha sido diseñado para soportar gran variedad de representaciones y herramientas analíticas para datasets complejos. También dispone de un *framework* para visualización con diferentes *layouts*.
- **Mahout**[4]: La librería de *machine learning* Apache Mahout™[50] esta diseñada para construir librerías escalables para *machine learning*. Provee de varias herramientas de *machine learning* para clustering, clasificación y filtrado colaborativo, implementadas sobre Apache Hadoop[5] utilizando el paradigma *map-reduce* [15].
- **Matlab** [6]: MATLAB®[61] es un lenguaje de alto nivel con un entorno interactivo para la computación numérica, visualización y programación. Posee varias herramientas para analizar datos, desarrollar algoritmos y crear modelos y aplicaciones.
- **Octave** [7]: Octave [19] es un lenguaje interpretado de alto nivel para la computación numérica. Proporciona extensas capacidades gráficas para la visualización y manipulación de datos. El lenguaje Octave es similar a Matlab, por lo que muchos programas son fácilmente portables.
- **R** [8]: R [53] es un lenguaje y un entorno para computación estadística y gráfica. Es parte del proyecto GNU. Proporciona varias técnicas estadísticas (modelado, test estadísticos, análisis de series temporales, clasificación, clustering, etc) y gráficas, es altamente extensible.

[1] `https://gephi.org`
[2] `http://www.graphviz.org`
[3] `http://jung.sourceforge.net`
[4] `http://mahout.apache.org`
[5] `http://hadoop.apache.org/`
[6] `http://www.mathworks.co.uk/products/matlab/index.html`
[7] `http://www.gnu.org/software/octave`
[8] `http://www.r-project.org`

- **Weka** [9]: Weka [27] es una colección de algoritmos de *machine learning* para tareas de *data mining*. Contiene herramientas para el pre-procesado de datos, clasificación, regresión, clustering, reglas de asociación y visualización. Es software *open source* bajo la *GNU General Public License*.

Otras herramientas que ayudan en el análisis social desde una perspectiva semántica son las siguientes:

- **Maui** [10]: Maui [43] es un programa que identifica de manera automática el tema principal de documentos. Para ello hace uso de *tags*, *keywords*, *keyphrases*, descriptores, indexación de términos o títulos de artículos en Wikipedia entre otros. También puede ser utilizado como extractor de terminologías e indexador de *topics* (temas) semi-automático.
- **Relfinder** [11]: Relfinder [39] es una herramienta de visualización de grafos, donde se representan las relaciones existentes entre dos términos. Esta basado en el *framework* de *open source* Adobe Flex y funciona con cualquier dataset de RDFs estandarizado para proporcionar acceso mediante consultas SPARQL.
- **Wikipedia Miner** [12]: Wikipedia Miner [47] es un complejo de herramientas diseñadas para aprovechar la información latente en Wikipedia mediante análisis semántico. Provee acceso a la estructura de Wikipedia, así como técnicas de medición para ver cómo están relacionados entre sí diversos términos o eliminar la ambigüedad de algunos de estos.

[9] `http://www.cs.waikato.ac.nz/ml/weka/`
[10] `https://code.google.com/p/maui-indexer/`
[11] `http://www.visualdataweb.org/relfinder.php`
[12] `http://wikipedia-miner.cms.waikato.ac.nz/`

3
Tweet Miner

En este capítulo se muestra el modelo implementado en el trabajo **TweetSemMiner: A Meta-Topic Identification Model for Twitter Using Semantic Analysis** [**45**], así como el nuevo modelo con el que se comparará en el próximo apartado de una forma más específica. En ambos casos, el objetivo es realizar un analizador semántico que nos permita identificar una serie de textos que se puedan englobar en un contexto determinado (*meta-topic*). Dado un nombre de usuario de *Twitter* (de perfil público), se extraen y analizan cierto número de *tweets* de cada uno de los usuarios a los que sigue, se llevan a cabo diversas operaciones mediante consultas SPARQL y el uso de algoritmos de similitud semántica (como LSA o NGD según el modelo respectivamente), obteniendo así un ranking con los *tweets* ordenados según su cercanía al *meta-topic* deseado.

En ambos modelos se ha optado por implementar dicho sistema con un único *meta-topic*: **videojuegos**; Dejando para trabajo futuro el ampliar la lista de *meta-topics* posibles. En esta sección se explican tanto la **arquitectura**, como el **funcionamiento** de ambos modelos aplicados a este *meta-topic*.

3.1. Arquitectura

La arquitectura de ambos modelos está dividida en distintos módulos, tal y como se muestra en la **Figura 3.1**. La arquitectura general es similar, aunque se han mejorado los módulos relacionados con el análisis semántico:

de ambos modelos es idéntica, cambiando únicamente la implementación de alguno de sus módulos:

- **Extracción de los *tweets* (1)**: Extrae los *tweets* de los usuarios a los que sigue el sujeto y los almacena para su posterior preprocesado.
- **Preprocesado del texto (2)**: Toma los *tweets* extraídos y los preprocesa (eliminando caracteres que no sean ASCII, stopwords, etc).
- **Análisis semántico (3)**: En este módulo se aplican algoritmos cuyo objetivo es hallar relaciones semánticas entre las *keywords* obtenidas de los *tweets*. Este módulo cambia su implementación dependiendo del modelo con el que se esté trabajando.

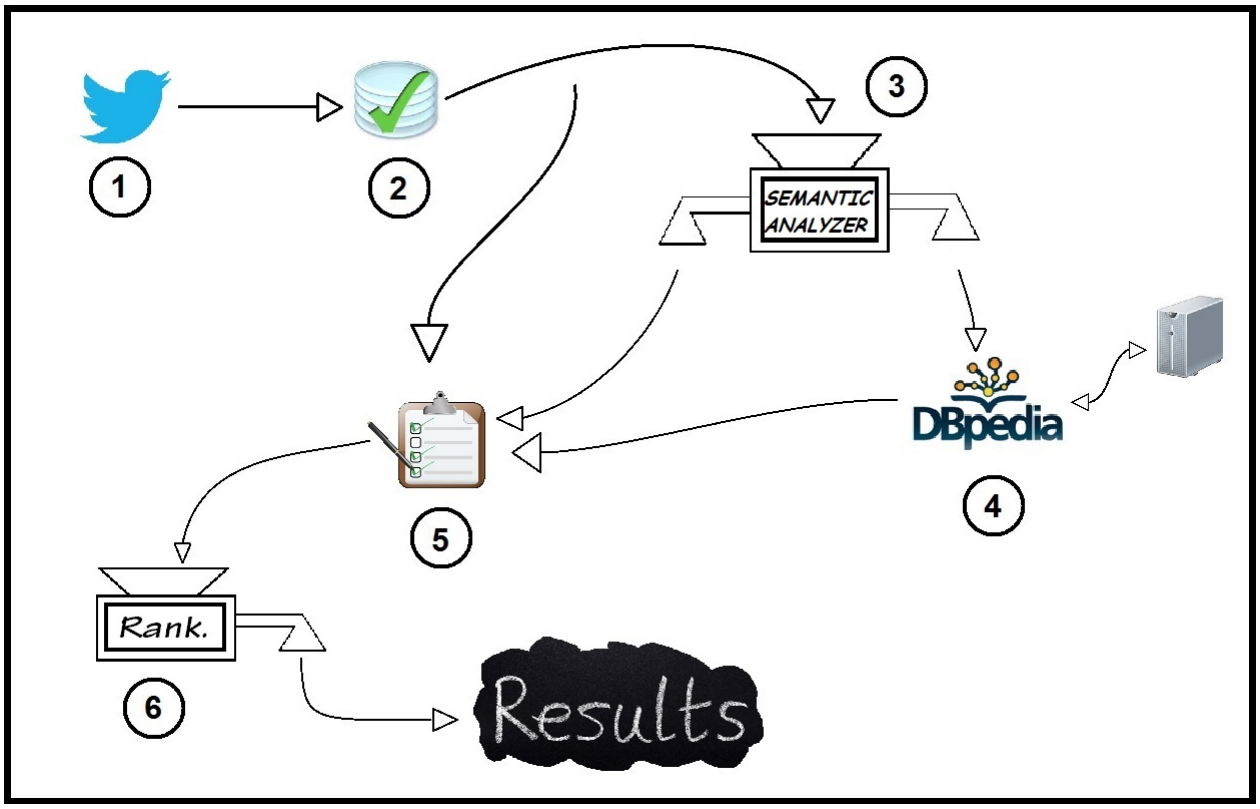

Figura 3.1: Esquema de la Arquitectura del primer modelo

- **Consultas a DBpedia mediante SPARQL (4)**: Recoge las *keywords* obtenidas mediante el análisis semántico y las utiliza para establecer relaciones con el *meta-topic* mediante consultas a DBpedia en formato SPARQL.
- **Valoración final de las *keywords* (5)**: En este módulo se juntan los valores obtenidos en el módulo (3) y de las consultas SPARQL, obteniendo así una valoración de cada *keywords* con respecto al *meta-topic* deseado. Este módulo cambia su implementación dependiendo del modelo con el que se esté trabajando.
- **Aplicación de ranking (6)**: Tomando la valoración obtenida del módulo anterior, se genera un valor para cada *tweet* a partir de sus *keywords* y el resultado obtenido se organiza de mayor a menor, obteniendo así un ranking que depende de la adecuación del *tweet* al *meta-topic*.

Prácticamente todos los módulos desarrollados en esta arquitectura son genéricos y se podrían utilizar para cualquier *meta-topic* excepto el módulo de **Consultas a DBpedia mediante SPARQL**, que requiere un tratamiento especial como se explica en la Sección 3.1.4.

3.1.1. Extracción de los tweets

La mayoría de redes sociales tienen una estructura compleja entre usuarios y la información que se puede obtener es muy densa y a menudo de poca utilidad, además la privacidad de dichas redes es siempre un tema conflictivo. No obstante *Twitter* provee de una gran cantidad de información que varía cada pocas horas y que al mismo tiempo no es ni demasiado breve ni demasiado extensa. Su política de privacidad se basa en el libre acceso a dicha información, es

por ello que se ha optado por utilizar *Twitter* como fuente de información a analizar en este trabajo.

En este módulo se toma el nombre del usuario de *Twitter* deseado (cuyo perfil debe ser público) y se obtienen los nombres de usuario de la gente a la que este sigue (apartado *following* de la cuenta del usuario). Posteriormente se recorre la lista de usuarios obtenidos y se extrae su *Time Line* o Línea Temporal, que representa las últimas publicaciones del usuario (incluyendo tanto *tweets* originales del usuario, como *retweets* que este haya llevado a cabo). Estos *tweets* se almacenan para su posterior procesado.

Para poder llevar a cabo estas operaciones de extracción *Twitter* proporciona una API[1] pública, que nos permite acceder a su corpus de datos, más concretamente utilizaremos su **REST API** que provee de las funciones primitivas de acceso a estados, timelines e información de los usuarios. No obstante, en este módulo se ha optado por utilizar una librería a más alto nivel denominada *Twitter4J*[2] que permite realizar las operaciones de búsqueda y extracción desde Java.

3.1.2. Preprocesado del texto

Los datos obtenidos en el módulo anterior pasan por un proceso de normalizado en el que se eliminan stopwords, URLs y elementos conflictivos (caracteres no ASCII entre otros). Tras este proceso vuelven a almacenarse para ser utilizados en el módulo siguiente.

3.1.3. Análisis Semántico

La implementación de este módulo varía según el modelo con el que estemos tratando:

En el trabajo **TweetSemMiner: A Meta-Topic Identification Model for Twitter Using Semantic Analysis [45]** se extraen las *keywords* de los textos y una vez extraídas se aplica LSA, que busca una relación semántica entre dichas *keywords*, obteniendo así una matriz término-término, la cual nos será útil a la hora de la evaluación, especialmente para relacionar aquellas *keywords* que no se encuentren en DBpedia.

LSA o *Latent Semantic Analysis* es un método matemático que trata de encontrar relaciones latentes dentro de una colección de documentos. En vez de analizar cada documento por separado, los analiza como un conjunto de términos e intenta encontrar relaciones entre ellos. A la hora de aplicar LSA, se ha optado por utilizado la librería *semanticpy*[3], que proporciona todo lo necesario para aplicar de forma sencilla y directa el algoritmo.

En el nuevo modelo, sin embargo, se ha optado por utilizar *NGD* [11] o **Normalized Google distance**; una medida de similitud semántica derivada del motor de búsqueda de *Google*. Se basa en que *kewwords* con un significado similar tienden a estar más *cerca* en unidades de distancia, mientras que palabras no similares tienden a estar apartadas entre si.

La distancia entre dos elementos (x, y) se define por *NGD* como:

$$NGD(x,y) = \frac{max(logf(x), logf(y)) - logf(x,y)}{logN - min(logf(x), logf(y))} \tag{3.1}$$

Donde $f(x)$ representa el número de páginas donde aparece x, $f(x,y)$ representa el número de páginas donde aparecen tanto x como y y N determina el número total de páginas sobre las que se está realizando la operación.

[1] `https://dev.twitter.com/docs/api/1.1`
[2] `http://twitter4j.org/en/index.html`
[3] `https://github.com/josephwilk/semanticpy`

Por ejemplo, si tenemos *gato*, *animal* y mil documentos en los cuales *gato* aparece 200 veces, *animal* aparece en 150 documentos y ambas aparecen juntas en 80, su distancia *NGD* sería:

$$NGD(gato, animal) = (log(200) - log(80))/(log(1000) - log(150)) = 0{,}48$$

Una vez aplicado *NGD* para cada par de *kewwords*, el módulo del nuevo modelo transforma la distancia obtenida en *similitud*; Para ello a cada par de claves le asignamos *1 - el valor obtenido mediante NGD*. Tras terminar estas operaciones se obtendrá una matriz término-término que se analizará en el quinto módulo del modelo.

3.1.4. Consultas a DBpedia mediante SPARQL

Con las *keywords* obtenidas en el módulo anterior se construyen consultas a un mirror de DBpedia[4]. Dicho mirror se encarga de, dada una *keyword*, buscar, y en caso de éxito, devolver la URL correspondiente a la *keyword* en DBpedia[5]. Con dicha URL el módulo hace uso de unas consultas SPARQL (diseñadas para crear relaciones entre los distintos niveles del árbol y las *keywords*). Si se encuentra relación entre la *keyword* y un término relacionado directamente con los videojuegos se le aplica una valoración positiva a dicha *keyword* (máximo valor 1). En caso contrario se le asigna el valor '0' a dicha *keyword*. Tras estas consultas se almacenan los resultados en formato [clave, valor] para su posterior utilización.

Al tratar información semi-estructurada como es el caso de DBpedia se ha decidido crear una semi-estructura en forma de árbol por cada *meta-topic* desarrollado. En la **Figura 3.2** podemos ver el árbol correspondiente al *meta-topic* (Videojuegos) tratado en este trabajo. Consultando distintos niveles del árbol nos permite dar una mayor o menor importancia a un término dentro del ámbito de los Videojuegos (dando mayor importancia a un término que hable de un videojuego sobre otro que hable de una empresa de videojuegos, por ejemplo).

Cabe destacar que este módulo es la única parte del modelo que habría que re-diseñar para hacer que el modelo actuase de acuerdo a un *meta-topic* distinto. Para ello, es clave desarrollar un nuevo árbol acorde con el nuevo *meta-topic* (analizando la información disponible de este para poder diseñar nuestra propia semi-estructura). Una vez desarrollado el nuevo árbol habría que re-definir las consultas SPARQL que comprueban si una *keyword* forma parte de nuestro árbol y en caso afirmativo, en que nivel se encuentran.

SPARQL (SPARQL Protocol and RDF Query Language) es un lenguaje de consultas que nos permite acceder a los RDF (Resource Description Framework) almacenados en DBpedia[6], obteniendo así información sobra las *keywords*. Dicha información, junto con el árbol de *meta-topic*, nos permite asignar un valor a cada *keyword*, como se ha explicado con anterioridad.

[4] `http://wiki.dbpedia.org/Lookup?v=1cn6`
[5] `http://dbpedia.org/resource/<key>`
[6] `http://dbpedia.org/`

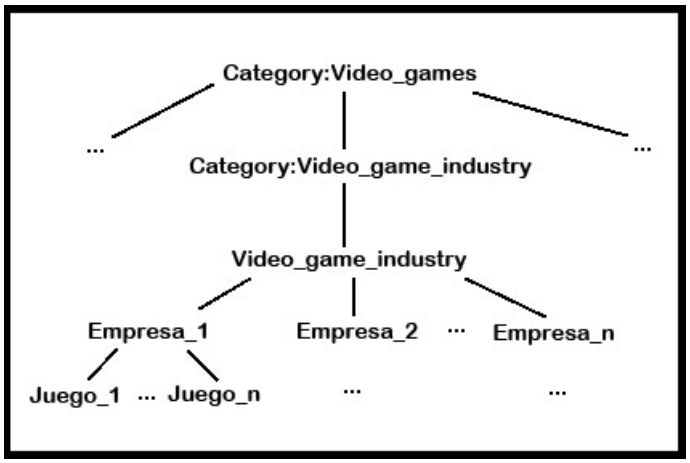

Figura 3.2: Estructura 'Videojuegos' en DBpedia

3.1.5. Valoración final de las keywords

En este módulo se complementan los resultados obtenidos de las consultas SPARQL con la relación término-término obtenida de ejecutar el módulo de análisis semántico, para realizar dicha fusión y obtener así una tabla de formato [clave, valor] con todas las *keywords* analizadas.

En el modelo presentado en **Modelo de Identificación de Meta-Topics a Través de Análisis Semántico de Conjuntos de Datos Extraídos de Twitter [45]** se hace uso del **Algoritmo 1** para llevar a cabo dicha operación.

Algorithm 1 Merge Stats Algorithm

```
1: Let key_value_table = sparql_key_value and set i = ∅
2: for row ∈ matrix do
3:     key = keys[i]
4:     j = ∅
5:     for col ∈ row do
6:         sub_key = keys[j]
7:         aux = eval(col) * sparql_key_value[key]
8:         if aux > key_value_table[key] then
9:             key_value_table[key] = aux
10:        end if
11:        j++
12:    end for
13:    i++
14: end for
```

Para ello primero se inicializa la tabla [clave, valor] con los valores de la tabla SPARQL (línea 1). Posteriormente por cada fila de la matriz término-término (línea 2) se obtiene la *keyword* de la primera posición (línea 3). Por cada posición de cada fila (línea 5) obtenemos también su *keyword*. Si el valor de dicha posición multiplicado por el valor original de dicha *keyword* en la tabla de SPARQL es mayor que el valor original de la tabla SPARQL para dicha *keyword*, le asignamos a dicha *keyword* el valor obtenido (líneas 7-10).

Algorithm 2 DBpedia Similarity Algorithm

1: Let $keys = getKeys()$
2: Let $matrix = getMatrix()$
3: Let $matrix = getMatrix()$
4: Let $threshold = getThreshold(keys, matrix)$
5: Let $clusters = getClusters(keys, matrix, threshold)$
6: Let $realClusters = getRealClusters(clusters, len(keys))$
7: Let $sparql = getSPARQLValues()$
8: Let $first = getSimilarityValues(sparql, keys, realClusters)$
9: Let $second = getSimilarityValues(first, keys, realClusters)$

El nuevo modelo por el contrario hace uso del **Algoritmo 2**. Los puntos clave del nuevo algoritmo son la generación de los clusters (líneas 4 a 6) y la aplicación de los valores por similitud (líneas 8 y 9). A continuación se explican de forma más detallada:

El primer paso para poder aplicar clustering en este módulo es definir un delimitador o threshold que permita agrupar las *keywords* en grupos. Para ello se ha optado por un threshold que cumpla que ningún elemento se quede sin cluster, es decir, se selecciona el mínimo valor de similitud del máximo de cada fila de la matriz de términos. Una vez obtenido el threshold se generan los clusters siguiendo las siguientes reglas (líneas 5 y 6):

- Para cada fila de la matriz se genera un cluster al cual se asignan todas las *keywords* cuyo valor sea superior al *threshold* obtenido.
- Si un cluster contiene todas las *keywords* se elimina.
- Si un cluster contiene a otro cluster por completo este último cluster es subsumido por el primero.

Por ejemplo, si tenemos la Tabla 3.1, los máximos valores de cada fila (sin incluir la diagonal) serían los presentados en la Tabla 3.2, de los cuales el mínimo sería *0.7* y por tanto sería el *threshold* elegido. A continuación se obtiene un cluster para cada fila de la matriz (aplicando el threshold seleccionado), obteniendo la Tabla 3.3. Se puede observar que los clusters C2 y C4 están contenidos en el C1, y el cluster C5 está contenido en C3, por lo que, finalmente, se obtendrían 2 clusters (C1 y C3).

E1	1	0.7	0.8	0.9	0.3
E2	0.7	1	0.6	0.1	0.2
E3	0.8	0.6	1	0.4	0.8
E4	0.9	0.1	0.1	1	0.3
E5	0.3	0.2	0.8	0.3	1

Tabla 3.1: Matriz de ejemplo

E1	0.99
E2	0.7
E3	0.8
E4	0.9
E5	0.8

Tabla 3.2: Máximo valor de cada fila de la matriz de ejemplo

C1	E1	E2	E3	E4
C2	E1	E2		
C3	E1	E3	E5	
C4	E1	E4		
C5	E3	E5		

Tabla 3.3: Clusters iniciales obtenidos de la matriz de ejemplo

La definición de la función getSimilarityValues está expuesta en el **Algoritmo 3**. En ella se puede ver (línea 2) como se toman solo las *keywords* con un valor superior a 0.4 (este valor es un delimitador que evita seleccionar demasiadas *keywords* de poco interés, lo cual ralentizaría la ejecución considerablemente; su valor es empírico). Una vez seleccionadas las *keywords* con un valor de interés, para cada *keyword* se busca, dentro del conjunto reducido, cuál es más similar a la misma (para ello utilizamos los clusters previamente generados). Una vez obtenido el elemento más similar al que estamos analizando comprobamos si **maxSimilarity * maxSimilarityValue** (línea 12) devuelve un valor superior al que tenía el elemento de por sí, en caso afirmativo el valor del elemento se sustituye por el nuevo valor obtenido. De este modo, las *keywords* que tenían coincidencias en DBpedia van propagando su valor a aquellos elementos más cercanos a ellos.

En el módulo implementado se suceden dos llamadas consecutivas a esta función; esto se debe a que en la primera pasada solo se propagan los valores de las *keywords* que tenían coincidencia directa con DBpedia, mientras que en la segunda pasada es posible propagar un nivel más dichos valores, dando así un peso a palabras relacionadas indirectamente con las *keywords* que tenían coincidencia directa.

Algorithm 3 getSimilarityValues

```
1:  Define similarityValues
2:  utilValues = getKeysThatMatter(keysValues, 0,4)
3:  for key_i ∈ enumerate(keys) do
4:      maxSimilarity = 0,0
5:      maxSimilarityValue = 0,0
6:      for key_j ∈ utilValues do
7:          auxSimilarity = getSimilarity(key_i, key_j, clusters)
8:          if auxSimilarity > maxSimilarity then
9:              maxSimilarity = auxSimilarity
10:             maxSimilarityValue = utilValues[key_j]
11:         end if
12:         similarityValue = maxSimilarity * maxSimilarityValue
13:         if similarityValue > prevValues[key_i] then
14:             similarityValues[key_i] = similarityValue
15:         else
16:             similarityValues[key_i] = prevValues[key_i]
17:         end if
18:     end for
19: end for
```

3.1.6. Aplicación de ranking

Una vez aplicada la valoración de las *keywords* se lleva a cabo la generación de la tabla [*tweet*, valor], donde, sumando los valores de cada *keyword* de un *tweet* obtenemos su valor.

Posteriormente se aplica una ordenación de mayor a menor, según el valor obtenido.

Aplicando métricas a dicha información se puede medir la eficiencia del análisis semántico realizado, que será comparado con el criterio que seguiría un ser humano.

3.2. Funcionamiento

En este apartado se definen las distintas conexiones que se llevan a cabo entre las distintas partes que conforman el modelo (tal y como se ve en la **Figura 3.3**).

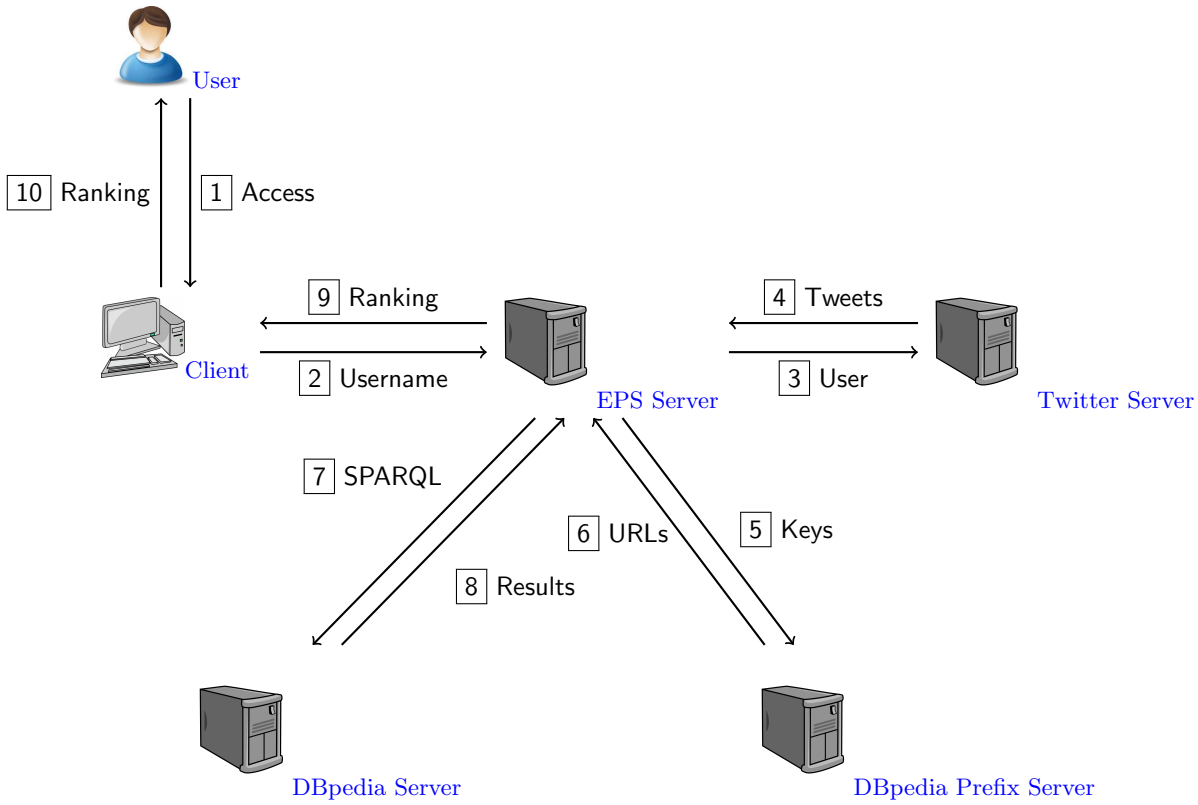

Figura 3.3: Proceso de creación del Árbol de Dependencias

Primero el usuario accede a la herramienta a través del cliente **(1)**. La herramienta envía la información del usuario al servidor **(2)**, donde a su vez envía peticiones a *Twitter* para extraer los *timeline* de los usuarios a los que sigue el usuario **(3)**. El servidor de *Twitter* devuelve los *tweets* correspondientes a dichos *timeline* **(4)** (20 *tweets* por timeline). El servidor obtiene las *keywords* de dichos *tweets*, aplica métricas de análisis semántico y envía peticiones de búsqueda a DBpedia Prefix Searcher **(5)**. En caso de encontrar coincidencias entre los términos y su BBDD **(6)** DBpedia devuelve una URL al servidor. Con dichas URL el servidor hace consultas SPARQL a DBpedia **(7)**, que devuelve los resultados obtenidos al servidor **(8)**. Con esta información y la obtenida del análisis semántico se puede hacer una valoración de la relación de cada término con el *meta-topic*. Una vez completadas las valoraciones por *keyword* se calcula el valor de cada *tweet* y a continuación se genera un ranking sobre el valor obtenido para los tweets **(9)**. Finalmente el cliente muestra este ranking **(10)** al usuario.

4
Pruebas

En este apartado se explican las pruebas realizadas para comprobar el correcto funcionamiento del nuevo modelo. Así mismo se explican también las pruebas realizadas para comparar cual de los dos modelos presenta un mejor funcionamiento (el modelo presentado en el trabajo **Modelo de Identificación de Meta-Topics a Través de Análisis Semántico de Conjuntos de Datos Extraídos de Twitter** [**45**] o el expuesto en este documento).

4.1. Pruebas del sistema completo

En la implementación de este modelo existen varias limitaciones (de servicio, memoria, red, etc), que han llevado a realizar una selección de parámetros para adecuar el modelo a un funcionamiento real. Esta sección explica dichos detalles módulo a módulo.

4.1.1. Extracción de los tweets y posterior preprocesado

En el nuevo modelo los dos primeros módulos de la arquitectura (extracción y preprocesado de los *tweets*) se han fusionado en un mismo programa. Este se encarga de extraer el timeline de los usuarios a los que sigue el usuario indicado (se ha utilizado el mismo usuario en ambos modelos para poder comparar de forma adecuada los resultados). La API de Twitter tiene bastantes restricciones a la hora de extraer información en grandes cantidades de sus datasets. La principal es que corta las consultas durante 15 minutos en caso de extraer demasiada información en un corto período de tiempo. Por ello, tras llevar a cabo diversas pruebas de estrés, se ha optado por restringir el número de usuarios de los que se extrae el timeline a un máximo de 50 por ejecución.

La elección del argumento de entrada es clara: el nombre de un usuario de Twitter válido y de perfil público. Con dicho nombre de usuario podemos acceder a sus datos de Twitter mediante los servicios que proporciona la API de Twitter y extraer los *tweets de la gente a la que sigue*. La siguiente restricción viene marcada de nuevo por la API, dado que sólo nos permite extraer un máximo de 20 *tweets* por cada timeline de un usuario.

Una vez que los datos han sido extraídos se preprocesan antes de ser almacenados en formato *CSV*. La versión implementada en este módulo elimina todo carácter no ASCII (salvo los

acentos), así como URLs y algunas stopwords, entre otros. Esta implementación es óptima para extraer información en inglés, pero plantea serias limitaciones en otros lenguajes, por lo que se recomienda trabajar con usuarios que comúnmente trabajen en dicho idioma.

4.1.2. Aplicación de NGD

Este módulo necesita los *tweets* pre-procesados del módulo anterior, por lo que la información obtenida en la ejecución del módulo anterior es su entrada.

Este módulo extrae las *keywords* de los *tweets* y genera una matriz de distancias entre términos (matriz término-término), por lo que la memoria y el tiempo de ejecución son las mayores limitaciones a enfrentar en este punto.

La limitación de usuarios sobre los que se extrae información y de *tweets* que se pueden extraer de los módulos anteriores permiten mantener estable el sistema, pero si se quieren aumentar se deberá desarrollar un sistema que optimice la arquitectura actual (como se verá en el apartado de *Trabajo Futuro*).

4.1.3. Consultas a DBpedia mediante SPARQL

Este módulo se divide en dos secciones:

La primera sección consulta al servicio Prefix Searcher de DBpedia para obtener una URL válida (en DBpedia) para cada *keyword*. Se debe respetar el formato de la consulta al servicio (evitar caracteres extraños, consultas vacías, etc).

Tras esto, la siguiente sección se encarga de realizar las consultas a DBpedia para decidir si una *keyword* pertenece o no al *meta-topic* con el que se está trabajando. Las limitaciones en este caso vienen enlazadas al diseño del árbol del *meta-topic*. A mayor complejidad en el árbol, mayor complejidad en las consultas y menor número de restricciones y viceversa.

4.1.4. Valoración final de las *keywords*

En este módulo se calcula un valor final de cercanía al *meta-topic* por cada *keyword*, el algoritmo utilizado para llevar a cabo esta valoración presenta una eficiencia de $O(n^2)$ que impone las restricciones a este módulo.

4.1.5. Aplicación de ranking

La única restricción a la hora de calcular el valor de cada *tweet* y generar el ranking es la restricción de eficiencia marcada por el algoritmo *sort* de ordenación.

4.2. Ejemplos reales

En este apartado se muestran *tweets* procesados por ambos modelos. Todos ellos han sido marcados como 'positivos' por los modelos. A continuación se muestra su auténtico valor (true positive — false positive), así como una breve explicación de como el modelo ha llevado a seleccionarlos como 'positivos'.

En la Tabla 4.1 pueden observarse los 5 *tweets* con un valor de ranking más elevado según el *Modelo A*, mientras que en la Tabla 4.2 podemos observar los 5 *tweets* más relevantes según el *Modelo B*.

Tweet	Valor real
RT @yuu_251: #GTA #GTAV #GTA5 #snapmatic #rockstargames @Rockstargames #GTAOnline #GTAPhotographers #PS4share	True Positive
RT @GTAsnapmatichub: Presenting some snaps by @laurent_777 #gta5 #Snapmatic #Gta #Rockstargames #GTAPhotographers #gtav #Snapmatichub	True Positive
RT @IEM: Congratulations to INnoVation your #IEM gamescom champion in @StarCraft II	True Positive
RIP to Sean P.... So sad man. He was a huge inspiration to me. Grew up on BCC. Prayers to his family, friends and Duck Down	False Positive
RT @btzyt: Magnum, P.I. starring Trevor Philips. @StevenOgg surely would be a great Magnum via @YouTube #GTAV	True Positive

Tabla 4.1: Modelo A. Ejemplos reales

Tweet	Valor real
RT @GTAsnapmatichub: Presenting some snaps by @laurent_777 #gta5 #Snapmatic #Gta #Rockstargames #GTAPhotographers #gtav #Snapmatichub	True Positive
RT @yuu_251: #GTA #GTAV #GTA5 #snapmatic #rockstargames @Rockstargames #GTAOnline #GTAPhotographers #PS4share	True Positive
RT @TeamSanshee: Today's packaging doodle theme: Adorable #DragonAge #IronBull.	True Positive
¡Estas son las personas ganadoras del sorteo #ESLCinesa! ¡Felicidades y gracias a todos y todas! #CounterStrike	True Positive
RT @ESL: The third season of #WCS Premier League is upon us! Who will rise among the world's best @StarCraft II players?	True Positive

Tabla 4.2: Modelo B. Ejemplos reales

Tanto en la Tabla 4.1 como en la Tabla 4.2 podemos observar que los dos primeros *tweets* seleccionados se repiten. Esto es debido a que ambos *tweets* presentan una gran cantidad de palabras que aparecen en DBpedia (Rockstargames, GTA, GTAV) y por tanto son fácilmente clasificables por ambos modelos.

En la Tabla 4.1 vemos que entre los 5 primeros *tweets* hay un falso positivo: la muerte de un actor de la cadena de televisión británica *BBC*. Este falso positivo se debe a la aplicación de LSA y la falta de textos de la misma temática entre los usuarios a los que sigue el usuario de prueba. LSA da un valor a cada *keyword*. Este valor es muy dependiente de la cantidad de textos a analizar, ya que si existen pocos textos que hablen de lo mismo, acabaran siendo relacionados erronamente con otros textos que bi hablen del meta-topic deseado, ya que prima la relación más fuerte y palabras como 'family', 'friend', etc pueden acabar relacionando el texto de forma equivocada.

En la Tabla 4.2 sin embargo vemos como un docuemento con una única palabra que aparece en DBpedia (DragonAge) obtiene un valor de ranking elevado. Esto es debido a que tanto el usuario (TeamSanshee) como uno de los personajes del juego DragonAge (IronBull) aparecen normalmente junto a dicho videojuego, por lo que en la agrupación por clustering heredan parte

del valor de DragonAge, dandole una mayor importancia al texto.

4.3. Comparativa entre modelos

En este apartado se hace referencia al modelo presentado en **Modelo de Identificación de Meta-Topics a Través de Análisis Semántico de Conjuntos de Datos Extraídos de Twitter [45]** como *Modelo A* y al presentado en este documento como *Modelo B*.

Se han tomado los rankings de *tweets* obtenidos de ejecutar tanto el *Modelo A*, como el *Modelo B* y se han seleccionado los primeros *65 tweets* de ambos para calcular sus *curvas ROC* (Receiver Operating Characteristic). Una *curva ROC* consiste en una representación gráfica de la tasa de acierto frente a la tasa de falsos positivos (resultados clasificados como positivos no siéndolos) para los diferentes puntos de corte (cutpoints) posibles de una prueba diagnóstica (prueba que hace uso de un modelo como clasificador).

Dado que tanto el *Modelo A* como el *Modelo B* calculan el valor de cada *tweet* de manera distinta, no se pueden comparar directamente los valores obtenidos en estos para comprobar cual de los dos funciona de forma más eficaz, por ello las curvas *ROC* son la elección ideal para comparar ambos modelos.

Analizar las *curva ROC* proporciona herramientas para descartar modelos subóptimos de una forma muy relacionada con el análisis de *coste/beneficio* en una toma de decisiones diagnóstica. Al analizar una *curva ROC* se deben tener en cuenta los siguientes puntos:

- Cuanto mayor sea la sensibilidad menor será la especificidad; donde la sensibilidad es la probabilidad de clasificar correctamente a un elemento positivo y la especificidad es la probabilidad de clasificar correctamente un elemento negativo. El valor de los falsos positivos es 1 menos la especificidad.
- Cuanto más cercana sea la curva a la diagonal de 45º del espacio *ROC* menos exacta será la prueba.
- Una curva *ROC* por encima de la diagonal de 45º expresa unos resultados positivos de predicción en la prueba, mientras que si la curva se muestra por debajo de la diagonal los resultados de la prueba presentan un pobre poder de predicción.

Rank Value	True Positive	False Positive
3+	2	0
2.5-3	1	0
2-2.5	1	1
1.5-2	9	8
1-1.5	17	26

Tabla 4.3: Modelo A. Muestra *tweets* positivos y falsos positivos

Las tablas *4.3* y *4.4* presentan los *tweets* marcados correctamente de forma positiva y los marcados erronamente de forma positiva dentro de un rango de valores del ranking. En las tablas *4.5* y *4.6* sin embargo se han normalizado los valores. Los *True Positive* se calculan dividiendo los clasificados correctamente como positivos hasta el *cutpoint* seleccionado, entre el total de positivos y los *False Positives* de forma análoga se calcula dividiendo los falsos positivos acumulados hasta el *cutpoint* entre el total de negativos.

Al comparar las curvas *ROC* del *Modelo A* (rojo) y el *Modelo B* (verde) en la *Figura 4.1* vemos como el nuevo modelo presenta una curva con un mayor área entre la misma y

Rank Value	True Positive	False Positive
8+	2	0
7-8	0	0
6-7	5	0
5-6	2	0
4-5	6	2
3-4	18	4
2.5-3	15	11

Tabla 4.4: Modelo B. Muestra *tweets* positivos y falsos positivos

Cutpoint	True Positive	False Positive
3	0,06	0,00
2.5	0,1	0,00
2	0,13	0,02
1,5	0,43	0,26
1	1,00	1,00

Tabla 4.5: Modelo A. Muestra porcentaje (de 0 a 1) de positivos y falsos positivos

Cutpoint	True Positive	False Positive
8	0,04	0,00
7	0,04	0,00
6	0,15	0,00
5	0,19	0,00
4	0,31	0,12
3	0,69	0,35
2.5	1	1

Tabla 4.6: Modelo B. Muestra porcentaje (de 0 a 1) de positivos y falsos positivos

la diagonal. Como se ha expresado con anterioridad, esto denota una mayor probabilidad del modelo de clasificar de forma correcta los *tweets*.

El usuario de prueba intenta imitar el comportamiento de un usuario medio, por ello el elenco de cuentas de *Twitter* a la que sigue es variado (series de televisión, deportistas, Twitter DEV, cuentas graciosas, cines, empresas de videojuegos, amigos, etc). El uso de *LSA* en el *Modelo A* impone que para llevar a cabo el análisis semántico sean necesarios una gran cantidad de documentos que hablen sobre el mismo *Meta-topic*, dado que, si no, las relaciones semánticas que encuentre pueden ser erróneas o incluso pueden pasar desapercibidas en el análisis. Sin embargo el *Modelo B*, haciendo uso de la métrica *NGD* y un sencillo algoritmo de clustering no tiene problemas en encontrar las relaciones semánticas presentes en los documentos, aunque estos sean un número reducido. El *Modelo A* presenta una curva *ROC* positiva, pero cuanto más variopinto sea un usuario más problemas tendrá para ejecutar su cometido de forma eficaz. Con el nuevo *Modelo B* no sólo se consigue mejorar la probabilidad de acierto frente a la de error con respecto a su predecesor (como puede verse en la *Figura 4.1*), si no que es capaz de hacerlo de forma constante independientemente de la cantidad de documentos con los que trabaje.

Si se desease un modelo que sólo mostrase *tweets* asociados al *meta-topic* (es decir, que eliminase todos los Falsos Positivos), se podría generar un buen filtro, aunque para ambos modelos se perderían muchos de los datos marcados en la franja de 0,0 y 0,2 en el eje y (true positives).

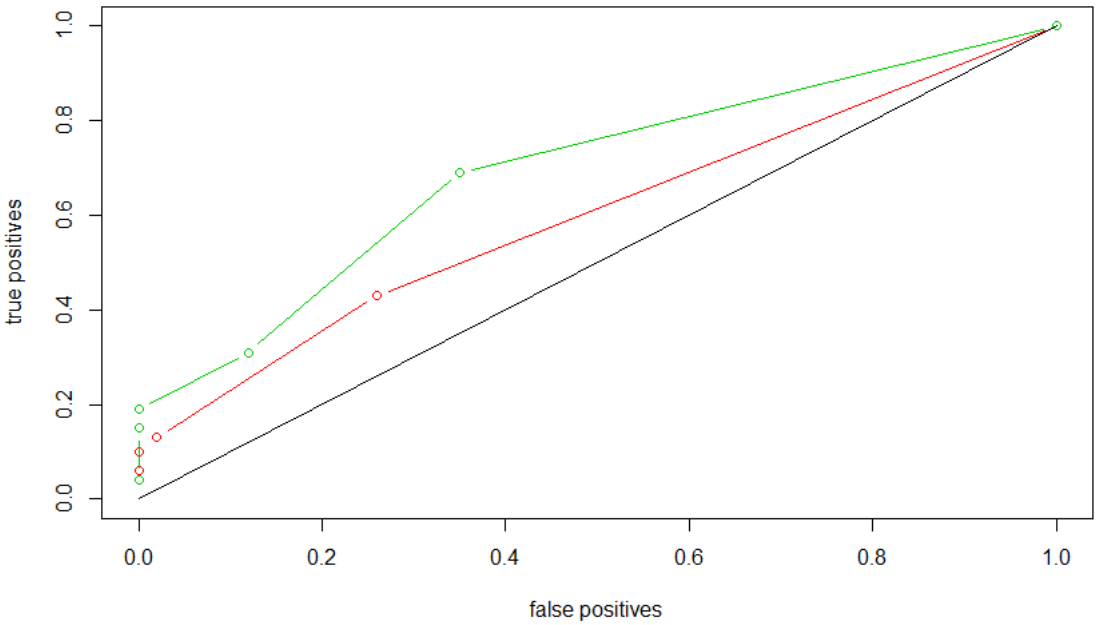

Figura 4.1: Modelo A. Curva ROC

5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

En el trabajo **TweetSemMiner: A Meta-Topic Identification Model for Twitter Using Semantic Analysis** [45] se presentaba un modelo donde se extraían datos provenientes de Twitter para posteriormente ser analizados semánticamente en pos de poder relacionarlos con un *meta-topic* preestablecido:"videojuegos". Para llevar a cabo el análisis semántico en dicho modelo se procesaban los datos extraídos de Twitter obteniendo así las *keywords*. Con dichas *keywords* se realizaban consultas a DBpedia para obtener relaciones entre los términos y el *meta-topic*. Al mismo tiempo, mediante LSA se generaba una matriz término-término entre todas las *keywords*, donde quedaban recogidas las relaciones semánticas de las mismas. Con la información extraída de estas técnicas se asignaba un valor a cada *tweet* extraído y se generaba un ranking con respecto a la relevancia de los mismos para el *meta-topic* seleccionado.

El objetivo de este trabajo consistía en tomar las bases del anterior modelo y evolucionar el análisis semántico, de forma que se mejorasen los resultados obtenidos. Para ello se han utilizado nuevas métricas como *NGD*, que, junto con técnicas de clustering han permitido que las *keywords* que poseen relación semántica con aquellas *keywords* con coincidencian en DBpedia. Una vez establecidas estas relaciones las *keywords* han podido 'heredar' un valor por cercanía al *meta-topic*. Una vez desarrollado el nuevo modelo se han llevado a cabo pruebas para comparar tanto la eficiencia del antiguo como del nuevo modelo.

Tras llevar a cabo ambas implementaciones y su posterior experimentación con datos reales en ambos modelos se han encontrado diversas limitaciones tales como:

- Restricciones en el volumen de datos que se puede extraer de la API Twitter.
- Restricciones en el acceso a datos que se puede consultar en la API Twitter.
- Restricciones en el volumen de información que se puede manejar al mismo tiempo debido a las restricciones Hardware de los equipos utilizados.
- Restricciones en el tiempo de ejecución de ambos modelos debido principalmente al desarrollo secuencial de la arquitectura.

Una vez realizados ambos desarrollos y su posterior comparación se ha obtenido la siguiente información:

1. Ambos modelos funcionan de forma eficaz si el usuario sigue a gente que no habla nada sobre el *meta-topic* seleccionado.
2. Ambos modelos funcionan de forma eficaz si el usuario sigue a gente que habla en gran medida sobre el *meta-topic* seleccionado.
3. El modelo antiguo presenta limitaciones en los casos en los que el usuario sigue a gente que hable de distintos *meta-topic*. Esto provoca que la información extraída sea insuficiente para obtener buenos resultados de las consultas de DBpedia y, por lo tanto, se vuelve más complicado el análisis de los datos.
4. El nuevo modelo funciona de forma eficaz independientemente de la gente a la que el usuario siga. Esto es debido al nuevo diseño en el que se da prioridad al valor obtenido de DBpedia que posteriormente se propaga por similitud, gracias a las técnicas de clustering aplicadas.

Tras analizar esta información se puede afirmar, no sólo que el nuevo modelo es más eficaz que el anterior (dado que mantiene su eficacia independientemente de la gente a la que siguen los usuarios), si no que la clave del proyecto pasa por potenciar las consultas a DBpedia. Son las *keywords* encontradas en el diccionario las que nos dan un mayor valor a la hora de detectar el *meta-topic* de un texto. Por ello cuanto mejor estén desarrolladas las consultas SPARQL, mejores resultados se hallarán a la hora de consultar a DBpedia y más acertados serán los resultados generados por el modelo.

El análisis semántico no debe ser olvidado, pues es el encargado de propagar los valores encontrados en DBpedia, de modo que aunque una *keyword* no aparezca en el diccionario pueda ser evaluada gracias a su cercanía semántica con otras *keyword* que si aparezcan en DBpedia.

5.2. Trabajo Futuro

Como ya se ha mencionado con anterioridad en este trabajo se trata de forma parcial con un proyecto de gran envergadura y por tanto existe una gran cantidad de trabajo a tratar en el futuro. Algunos de los puntos más importantes a tratar se exponen a continuación:

- Rediseñar la arquitectura del modelo para optimizar todos los procesos llevados a cabo.
- Se podría mejorar el árbol utilizado para realizar las consultas SPARQL. Cuantos más niveles consigan definirse será más fácil obtener información útil de DBpedia.
- Siguiendo el punto anterior sería interesante intentar diseñar una forma de obtener los árboles de las queries de forma automatizada, sin requerir un estudio detallado por parte de los desarrolladores del mismo.
- Podría implementarse una nueva aproximación donde se contemplasen más *meta-topics*, de forma que el usuario pudiese elegir en cada momento sobre cual realizar el análisis.
- Podrían llevarse a cabo pruebas más extensas y probar distintas métricas en los algoritmos y en la evaluación para poder analizar qué conclusiones se obtienen de forma más amplia.
- Se podría considerar que el modelo trabajase con un mayor número de idiomas, de modo que pudiésemos analizar datos procedentes de distintas regiones del mundo.

- Sería interesante extraer una mayor cantidad de datos por cada *tweet*, como su id, si es o no un *retweet*, si es una respuesta a otro comentario, etc. Esto facilitaría más información social dentro del análisis a nivel de red.
- El modelo podría implementarse como un servicio RESTful para poder realizar consultas a través de clientes independientes, que además pudiesen aportar datos nuevos a la red.

Glosario de acrónimos

- **API:** Application Programming Interface
- **ASCII:** American Standard Code for Information Interchange
- **BBDD:** Base de Datos
- **CPM:** Clique Percolation Method
- **EBC:** Edge Betweenness Centrality
- **EM:** Expectation - Maximization
- **HITS:** Hyperlink-Induced Topic Search
- **JUNG:** Java Universal Network/Graph Framework
- **KNN:** K-Nearest Neighbour
- **LDA:** Lineal Discriminant Analysis
- **LOPD:** Ley Oficial de Protección de Datos
- **LSA:** Latent Semantic Analysis
- **NB:** Naive Bayes
- **NGD:** Normalized Google distance
- **PCA:** Principal Component Analysis
- **RESTful:** Representational State Transfer
- **RDF:** Resource Descrition Framework
- **ROC:** Receiver Operating Characteristic
- **SPARQL:** SPARQL Protocol and RDF Query Language
- **SQL:** Structured Query Language
- **SVM:** Support Vector Machines
- **TDT:** Topic Detection and Tracking
- **URL:** Uniform Resource Locator

Bibliografía

[1] CharuC. Aggarwal and Haixun Wang. Text mining in social networks. In Charu C. Aggarwal, editor, *Social Network Data Analytics*, pages 353–378. Springer US, 2011.

[2] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[3] L. Backstrom and J. Leskovec. Supervised Random Walks: Predicting and Recommending Links in Social Networks. November 2010.

[4] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, October 1999.

[5] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. 2009.

[6] Michael Berry. *Survey of Text Mining : Clustering, Classification, and Retrieval*. Springer, September 2003.

[7] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97:245–271, December 1997.

[8] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[9] Susan Rovezzi Carroll and David J. Carroll. *Statistics Made Simple for School Leaders*. Rowman & Littlefield, 2002.

[10] Salvatore A. Catanese, Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Alessandro Provetti. Crawling facebook for social network analysis purposes. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, WIMS '11, pages 52:1–52:8, New York, NY, USA, 2011. ACM.

[11] Andrew R. Cohen and Paul M. B. Vitányi. Normalized google distance of multisets with applications. *CoRR*, abs/1308.3177, 2013.

[12] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[13] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21 –27, january 1967.

[14] Leticia Curiel, Bruno Baruque, Carlos Dueñas, Emilio Corchado, and Cristina Pérez-Tárrago. Genetic algorithms to simplify prognosis of endocarditis. In *Proceedings of the 12th international conference on Intelligent data engineering and automated learning*, IDEAL'11, pages 454–462, Berlin, Heidelberg, 2011. Springer-Verlag.

[15] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[16] K. Delac, M. Grgic, and S. Grgic. Independent comparative study of PCA, ICA, and LDA on the FERET data set. *International Journal of Imaging Systems and Technology*, 15(5):252–260, 2005.

[17] Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique Percolation in Random Networks. *Physical Review Letters*, 94(16):160202–1 – 160202–4, Apr 2005.

[18] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Mach. Learn.*, 29(2-3):103–130, November 1997.

[19] John W. Eaton. *GNU Octave Manual*. Network Theory Limited, 2002.

[20] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz - Open Source Graph Drawing Tools. *Graph Drawing*, pages 483–484, 2001.

[21] P. Erdős and A. Rényi. On random graphs. I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[22] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, September 2004.

[23] Emilio Ferrara. A Large-Scale Community Structure Analysis In Facebook, March 2012.

[24] Santo Fortunato, Vito Latora, and Massimo Marchiori. Method to find community structures based on information centrality. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 70(5):056104, 2004.

[25] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826, June 2002.

[26] W. Graham. *Facebook API Developers Guide*. Apresspod Series. Apress, 2008.

[27] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[28] Jiawei Han and Micheline Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.

[29] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[30] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why We Twitter: An Analysis of a Microblogging Community. In Haizheng Zhang, Myra Spiliopoulou, Bamshad Mobasher, Giles, Andrew McCallum, Olfa Nasraoui, Jaideep Srivastava, and John Yen, editors, *Advances in Web Mining and Web Usage Analysis*, volume 5439 of *Lecture Notes in Computer Science*, chapter 7, pages 118–138. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[31] Keehyung Kim, RI (Bob) McKay, and Byung-Ro Moon. Multiobjective evolutionary algorithms for dynamic social network clustering. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 1179–1186, New York, NY, USA, 2010. ACM.

[32] Jon M. Kleinberg. Hubs, authorities, and communities. *ACM Comput. Surv.*, 31(4es), December 1999.

[33] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97:273–324, December 1997.

[34] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 591–600, New York, NY, USA, 2010. ACM.

[35] Daniel T. Larose. *Discovering Knowledge in Data.* John Wiley and Sons, 2005.

[36] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 1:1–29, 2012.

[37] Kevin Lewis, Jason Kaufman, Marco Gonzalez, Andreas Wimmer, and Nicholas Christakis. Tastes, ties, and time: A new social network dataset using Facebook.com. *Social Networks*, 30(4):330–342, October 2008.

[38] Marek Lipczak and Evangelos Milios. Agglomerative genetic algorithm for clustering in social networks. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 1243–1250, New York, NY, USA, 2009. ACM.

[39] Steffen Lohmann, Philipp Heim, Timo Stegemann, and Jürgen Ziegler. The relfinder user interface: Interactive exploration of relationships between objects of interest. In *Proceedings of the 14th International Conference on Intelligent User Interfaces (IUI 2010)*, pages 421–422, New York, NY, USA, 2010. ACM.

[40] D.G. Lowe. Local feature view clustering for 3d object recognition. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–682–I–688 vol.1, 2001.

[41] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[42] Kevin Makice. *Twitter API: Up and Running: Learn How to Build Applications with the Twitter API.* O'Reilly Media, Inc., 1 edition, April 2009.

[43] Olena Medelyan, Vye Perrone, and Ian H. Witten. Subject metadata support powered by maui. In *Proceedings of the 10th Annual Joint Conference on Digital Libraries*, JCDL '10, pages 407–408, New York, NY, USA, 2010. ACM.

[44] Brendan Meeder, Brian Karrer, Amin Sayedi, R. Ravi, Christian Borgs, and Jennifer Chayes. We know who you followed last summer: inferring social link creation times in twitter. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 517–526, New York, NY, USA, 2011. ACM.

[45] HéctorD. Menéndez, Carlos Delgado-Calle, and David Camacho. Tweetsemminer: A meta-topic identification model for twitter using semantic analysis. In Emilio Corchado, JoséA. Lozano, Héctor Quintián, and Hujun Yin, editors, *Intelligent Data Engineering and Automated Learning – IDEAL 2014*, volume 8669 of *Lecture Notes in Computer Science*, pages 69–76. Springer International Publishing, 2014.

[46] George A. Miller. WordNet: A Lexical Database for English. *Commun. ACM*, 38(11):39–41, November 1995.

[47] David Milne and Ian H. Witten. An open-source toolkit for mining wikipedia. *Artificial Intelligence*, 194(0):222 – 239, 2013.

[48] G. Nathiya, S. C. Punitha, and M. Punithavalli. An analytical study on behavior of clusters using k means, em and k* means algorithm. *CoRR*, abs/1004.1743, 2010.

[49] J. O'Madadhain, D. Fisher, S. White, and Y. Boey. The JUNG (Java Universal Network/Graph) Framework. Technical report, UCI-ICS, October 2003.

[50] Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. *Mahout in Action*. Manning Publications, 1 edition, January 2011.

[51] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF – W3C recommendation. Technical report, W3C, 2008.

[52] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[53] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010.

[54] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Phys. Rev. E*, 74:016110, Jul 2006.

[55] Volker Roth and Tilman Lange. Feature selection in clustering problems. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16, pages 1–8. MIT Press, Cambridge, MA, 2004.

[56] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):824–840, May 2009.

[57] Yuri Takhteyev, Anatoliy Gruzd, and Barry Wellman. Geography of Twitter networks. *Social Networks*, 34(1):73–81, January 2012.

[58] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):409–10, 1998.

[59] ShireenM. Zaki and Hujun Yin. A semi-supervised learning algorithm for growing neural gas in face recognition. *Journal of Mathematical Modelling and Algorithms*, 7(4):425–435, 2008.

[60] Waigai Zhen. *Graph Theory and its Engineering Applications.* Advanced series in electrical and computing engineering. World Scientific Publishing Company, Incorporated, 1997.

[61] Dongli Zhou, Wesley K. Thompson, and Greg Siegle. MATLAB toolbox for functional connectivity. *NeuroImage*, 47(4):1590–1607, October 2009.